SEDAR: A Semantic Data Reservoir for Heterogeneous Datasets

Sayed Hoseini
Hochschule Niederrhein
Krefeld, Germany
sayed.hoseini@hs-niederrhein.de

Ahmed Ali
RWTH Aachen University
Aachen, Germany
oraby.ali@rwth-aachen.de

Haron Shaker
RWTH Aachen University
Aachen, Germany
haron.shaker@rwth-aachen.de

Christoph Quix
Fraunhofer FIT & Hochschule Niederrhein
St. Augustin & Krefeld, Germany
christoph.quix@fit.fraunhofer.de

ABSTRACT

Data lakes have emerged as a solution for managing vast and diverse datasets for modern data analytics. To prevent them from becoming ungoverned, semantic data management techniques are crucial, which involve connecting metadata with knowledge graphs, following the principles of Linked Data. This semantic layer enables more expressive data management, integration from various sources and enhances data access utilizing the concepts and relations to semantically enrich the data. Some frameworks have been proposed, but requirements like data versioning, linking of datasets, managing machine learning projects, automated semantic modeling and ontology-based data access are not supported in one uniform system. We demonstrate *SEDAR*, a comprehensive semantic data lake that includes support for data ingestion, storage, processing, and governance with a special focus on semantic data management. The demo will showcase how the system allows for various ingestion scenarios, metadata enrichment, data source linking, profiling, semantic modeling, data integration and processing inside a machine learning life cycle.

ACM Reference Format:
Sayed Hoseini, Ahmed Ali, Haron Shaker, and Christoph Quix. 2023. SEDAR: A Semantic Data Reservoir for Heterogeneous Datasets. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3614753

1 INTRODUCTION

Data lakes (DLs) have been addressed intensively in research and practice in the recent years as an important element for data science and machine learning projects. DLs are scalable repositories to ingest raw data in its original format from heterogeneous data sources; thus, only a minimal effort is required for ingesting data into a DL which makes them an efficient tool to collect, store, link, and transform datasets. While first implementations aimed at processing 'big data' efficiently using distributed, scalable systems like *Hadoop*, the need for proper management of metadata and data quality in data lakes has also been recognized [11, 14, 15].

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614753

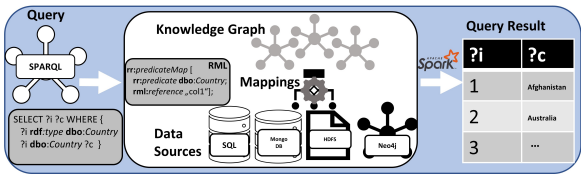


Figure 1: Query processing with KGs and semantic mappings

DLs can be applied in many applications, for example, management of heterogeneous data in Industry 4.0 [5] or in digital chemistry [4]. Data from various sources needs to be linked and enriched to be useful for a digital twin. In our use case of a smart paint factory [30], various datasets of experiments for paints need to be captured, cataloged and made available for machine learning (ML) [34]. Data sources are very heterogeneous, e.g., streams of machine sensors, interfaces of control software, databases and images of microscopes, and finally scripts and models from ML. As these external sources change in time, the system needs to track these continuously and efficiently.

Thus, meaningful metadata is essential for data usability, especially for users with limited domain knowledge or unfamiliarity with a dataset. Semantic information can enrich structural metadata by including contextual information that is not inherent to individual data sources. Knowledge graphs (KGs) are employed by organizations to capture domain knowledge, to describe raw data sources, and to improve data governance [9, 17]. Interpretability and interoperability of data is increased if semantically rich models are available as additional descriptive metadata. Semantic information expressed in ontologies has also proven useful for data integration using ontology-based data access (OBDA) [32]. Fig. 1 sketches the query processing in SEDAR. Queries are expressed in terms of the KG (an ontology) using the semantic query language SPARQL. Mappings are expressed in the RDF Mapping Language RML to link data sources to the KG. The query engine of SEDAR translates the SPARQL into queries for the underlying database systems and integrates the results. Thus, SEDAR allows to query heterogeneous data sources using a uniform semantic data model.

Our system is also able to perform OBDA, but the focus of this paper is on the challenging tasks of creating semantic models and mappings. According to [23], two important processes ensure the quality and acceptance of semantic data management (SDM): (1) creating and maintaining conceptualizations (i.e., in the form of KGs), and (2) describing data sources through semantic models,

which link data source schemas and conceptualizations. Creating such semantic models for a dataset involves understanding the existing data source, examining suitable KGs, and establishing links between raw data and concepts within the KG. While often knowledge and data management is done in separate systems, the main contribution of *SEDAR* is an implementation to bridge this gap.

In this paper, we present *SEDAR* as a comprehensive semantic DL that addresses the aforementioned requirements. The system is designed in a modular, easily extendable and flexible architecture based on microservices, making it suitable as a workbench for further research in DLs. It is built on existing open-source technologies in the area of big data management, but extends these components by the following more sophisticated functionalities: (i) Generic ingestion, provenance and different versions of datasets are supported by using an efficient delta store [3]. (ii) A semantic data catalog to manage datasets effectively for various use cases including data polymorphism, quality management, linkage, discovery, analytics, and MLOps. (iii) A semantic metadata management with an expressive GUI and ML-based techniques to generate accurate semantic models quickly. (iv) *SEDAR* implements a polystore with semantic query processing to query datasets uniformly on-demand without the need to know how the data is organized physically.

2 SYSTEM ARCHITECTURE

The core of *SEDAR* is composed of several state-of-the art big data technologies (e.g., *Spark*, *Hadoop*, *Kafka*), enriched with semantic metadata management (e.g., *Fuseki*, *WebVOWL*, *PLASMA*) and a modern, responsive user interface. The system architecture is illustrated in Fig. 2 and is based on a four layer architecture [18].

To realize a true polystore [20], the *storage layer* is composed of different storage systems to support efficient storage for heterogeneous data structures. Queries on heterogeneous data models (such as relations, JSON documents, or graphs) can be executed most efficiently in systems that are optimized on these structures (i.e., relational, document-oriented, or graph database systems). In the current version, we utilize *PostgreSQL* for structured relational data, *MongoDB* for nested documents (e.g., JSON and XML), and *Neo4j* for *SEDAR's* graph-based metadata model. KGs are maintained as domain-specific conceptualizations and are managed by *Fuseki* which supports typical Semantic Web languages and provides a SPARQL endpoint. Finally, *Hadoop* with its file system *HDFS* is used to store files with arbitrary structures.

The *transformation layer* implements various operations for data preparation, e.g., cleansing, transformations, or joins. We employ *Apache Spark* as a unified data transformation engine for large-scale data processing with implicit data parallelism and fault tolerance. *Spark*'s query engine supports heterogeneous data stores in the form of *DataFrames*; a uniform data representation for relational and nested data structures, which allows queries across different data stores. The Spark jobs are created by the backend from queries or data integration workflows defined using an intuitive user interface (UI) in the *interaction layer*. A programmatic interface to work with the data in *SEDAR* is provided by an integrated *Jupyter Notebook* environment (see Sec. 2.3).

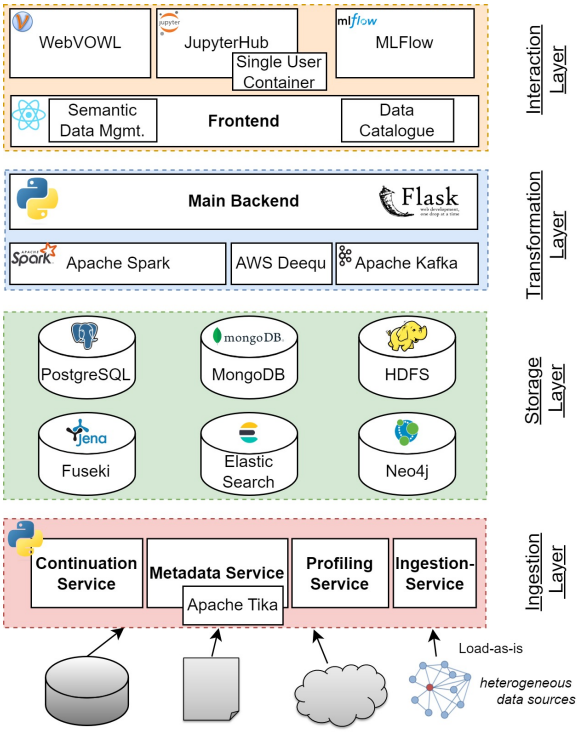


Figure 2: The *SEDAR* system architecture

2.1 Ingestion

To ensure generality and adherence to previous requirements, the development of the ingestion layer focused on five key aspects.

Firstly, the ingestion interface should be independent of any formats and data sources, requiring extension of *Spark* to support arbitrary custom configurations. A plugin mechanism was implemented to dynamically support custom ingestions which is triggered at two points: data loading to create the *DataFrame*, and post-loading transformation before storing it. Although a DL should usually store the data in its original format, this post-processing of loaded data is necessary for cases like ingesting streams from *Apache Kafka*, where the data is represented as a byte array. A plugin can be used for a syntactical transformation into a structured data format, such as a table or JSON document.

Second, the ingestion should not be limited to particular systems in the storage layer and allow the addition of further databases.

Moreover, it should be possible to ingest external sources continuously, i.e., data streams. *Kafka*, an open-source distributed event streaming platform and message-broker, registers and queues processing events and distributes messages between services.

Then, the ingestion must allow versioning, i.e., updates of datasets should be recognized and stored as deltas using change data capture (CDC) [21]. By doing so, the system can incrementally process changes instead of re-loading the dataset completely. For this purpose, we integrated the *Delta Lake* API [3]. Data is stored in *DeltaTables* that come with additional metadata. By default, the most recent version is used when reading, but one may also access previous versions. CDC is implemented in two ways: first, using an internal

solution based on the algorithm in [21] for (semi-)structured data; second, in order to transfer changes to a *DeltaTable*, the external data source might be able to provide the delta information explicitly, i.e., added, deleted and updated elements.

Finally, since data ingestion is the first phase of the data life-cycle, metadata must be collected in this first step. In *SEDAR*, the metadata service extracts metadata automatically and manages information about versions, updates, profiles, etc.

2.2 Data Catalog

To manage big data in industrial scenarios, a data catalog is fundamental for any organization. From related work [6, 10, 25, 26, 29, 35], we have identified several technical requirements for comprehensive metadata management: (1) indexing, (2) data linkage, (3) data polymorphism / multiple zones (e.g., different formats or degree of processing), (4) data versioning, (5) usage tracking, (6) granularity levels, (7) similarity relationships, (8) semantic enrichment, and (9) data quality management. Furthermore, the data catalog must ensure ease-of-use for non-technical users as well as provide means to manage access to sensitive information, i.e., privacy-preserving measures. These requirements can only be met partially by existing DL systems [25].

We introduce the concept of workspaces to the system which enables data security and supports data governance. Every user is assigned to a workspace and may add other users, but generally cannot access (meta-)data from other workspaces unless specific access is granted, or the workspace is made publicly available. For each dataset in a workspace, the system displays status and schema information as well as related datasets and they can be made publicly available or only to single users.

Dataset discovery supports various options: by utilizing the semantic information, by classical keyword-based search to match all metadata and source data indexed in *Elasticsearch* or by selecting further criteria, for example, status (raw / processed, private / public), time frames, owners and schema type (e.g., structured) and more. These results may be further exploited through links between datasets which are generated by multiple mechanisms: through the presence of primary- and foreign-key constructs, via history or lineage information and user-defined links or semantics.

The user interface for datasets is organized in multiple tabs. Queries may be specified in SparkSQL to query arbitrarily complex data structures directly without a semantic layer. Semantic query processing as illustrated in Fig. 1 is described in Sec. 3. The Profile tab displays the schema and statistical information of a dataset. It also allows to switch between versions of the same dataset and to open the data quality pipeline. We utilize Amazon's *Deequ* [27] library for automating the verification of data quality at scale combining common quality metrics with user-defined validation constraints to enable unit tests for data. In SEDAR, the data validation process is organized into a pipeline of profiling, constraint definition, anomaly detection, constraint suggestion, filtering and ingestion of the processed version. One may add further semantic annotation to each attribute, entity, or dataset; primary and foreign key relationships to attributes of the same or other datasets can be defined, and columns can be marked to contain sensitive information. The History tab displays info about the individual versions of a dataset and allows to perform updates (see Sec. 2.1). The Lineage tab provides info about previous transformations from which the dataset might have emerged. The Analytics tab allows to create computational notebooks with the corresponding code to access the dataset as a *DataFrame*.

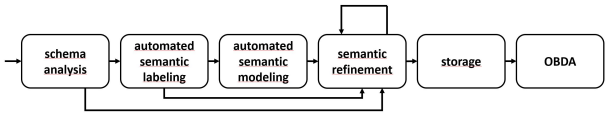


Figure 3: Semantic modeling pipeline [24] with OBDA

2.3 MLOps

DLs should provide data for machine learning (ML) applications. In our use case in the context of digital chemistry, SEDAR stores data from high-throughput experiments to optimize chemical formulations by using ML. To streamline the ML life cycle, MLOps is proposed [2]. In this regard, SEDAR employs *MLFlow* [33] to effectively manage the in- and outputs of ML applications. This includes tracking model versions, code and tuning parameters, ensuring result reproducibility, and facilitating model deployment.

We have deeply integrated *Jupyter Notebooks* into the interaction layer, which are hosted from a *JupyterHub* enabling users to have personalized environments for their work. The authentication between the various web services in *SEDAR* is centralized using the *OAuth* protocol and *JWT* tokens. Our self-hosted *GitLab* instance serves as the identity provider and also functions as a central repository for storing code.

Within *SEDAR*, users can select datasets, choose the desired learning method, and have the system automatically generate a notebook with the necessary code to first load the *DataFrames* from storage, then register their experiments, track individual runs and software environments, store model artifacts, and optionally deploy their applications via REST API exposure. While all model artefacts are stored in *HDFS*, *MLFlow*'s metadata is managed in a *PostgreSQL* database; the system establishes corresponding connections between workspaces, datasets, users and notebooks, thereby extending dataset search by additional filters.

3 SEMANTIC DATA MANAGEMENT

The process in SEDAR to enrich the data sources semantically and to map them to a KG is based on the semantic modeling pipeline presented in[24], shown in Fig. 3. The implementation is based on several existing components, but these components had several limitations that have been removed in SEDAR (e.g., applicability to arbitrary datasets and ontologies, use of standardized data formats to ensure interoperability between the components, improved usability of UI components).

While schema analysis is done during ingestion, for the automated assignment of semantic labels we selected to implement *Tab2KG* [13], due to its ability to interpret previously unseen data using domain-specific KGs. Semantic labels establish the connection between data sources and KG concepts, describing the semantics of associated data source attributes. Accurate semantic labels are

crucial for effective semantic modeling, forming elements of a preliminary semantic model. To facilitate semantic table interpretation, *Tab2KG* computes semantic profiles for domain ontologies and data tables. These profiles consist of feature vectors containing statistical measures whose selection is motivated by its expected effectiveness in semantic table interpretation. Unlike domain-specific approaches (e.g., [7, 31]), *Tab2KG* employs a Siamese Network that generalizes to unseen data by inducing a metric representing domain-independent similarity between unknown and known samples. The prediction of similarity between a column and a data type relation relies on the network's previous experience with similar profiles. Through a greedy approach, each table column is mapped to a data type relation based on a set of candidate column mappings and their corresponding similarity scores in the input data table.

The output of *Tab2KG* is utilized as input for the Semantic Modeling machIne (*SeMi* [12]) which uses graph neural networks (GNNs) to generate semantic models. By leveraging linked data repositories as background knowledge, the GNN learns latent features for entities and properties, encoding them in a vector space by utilizing the local neighborhood structures within the linked data graph. The processing pipeline involves creating an initial semantic model using a Steiner tree algorithm. Subsequently, the semantic model is refined by reconstructing the linked data edges based on the latent representation of entities and properties. The default model is based on the Relational Graph Convolutional Network [28] and the training is carried out through *PyKEEN* [1], a package for KG embeddings. *SeMI* enriches the semantic labels coming from the previous phase by additional meaningful concepts and provides plausible relationships that hold between them.

These previous automated approaches provide an initial semantic model to significantly reduce the overhead that comes with the time consuming creation of suitable mappings to data sources. However, incorrect or missing labels severely impact the quality of semantic models. Therefore, in the phase of semantic refinement, human operators must have the opportunity to manually enhance the quality of semantic labels and models by checking, validating, correcting, selecting, or exchanging concepts, as well as adding new ones. We integrated the open-source platform *PLASMA* [24] into *SEDAR*'s interaction layer to facilitate the visual management of semantic models. Users creating semantic labels can define and utilize concepts and relationships not previously known to the KG. This integration of novel knowledge into the ontology of the universal KG is particularly important when the desired concept is absent from the current conceptualizations. For *SEDAR*, we extended *PLASMA*'s capabilities to perform semantic data integration, i.e., modeling multiple datasets and unifying their representations grounded on the KGs registered in *SEDAR* and ported its metadata store to our internal *PostgreSQL*. From within *PLASMA*, users can trigger the automated creation of an initial semantic model, then load the output from *SeMI* and visualize it and finally refine it manually if desired.

The modeling tool PLASMA previously used a proprietary language for mappings, we adapted this in SEDAR to use the standard language RML [8]. This semantic description of the sources can be used for querying and to perform OBDA. OBDA provides a conceptual view of the data with which users can query various data sources without needing to know how the data is organized or where it is stored. The query is expressed in terms of the ontology. The query process was illustrated in Fig. 1: queries are written in SPARQL, evaluated against the *RML* mappings that establish the connection between concepts in the KGs and schema elements of the heterogeneous data sources. The query is translated into Spark-SQL and then *Spark* reads the relevant data and performs necessary transformations to return the query results in a *Dataframe*. To facilitate this type of semantics-based on-demand data access, we have incorporated *Squerall* [22], a scalable engine that enables querying multiple databases simultaneously. *Squerall* employs wrappers to directly query heterogeneous data sources in their original formats. To ensure compatibility with our PySpark implementation, we have translated certain parts of the Scala code into Python and adjusted other sections to align with our mapping serialization method. This method is based, in part, on the human-readable text-based representation for declarative Linked Data generation rules known as *YARRRML* [16], simplifying the readability of mappings drastically. *Squerall* effectively addresses the diversity challenge of Big Data by offering convenient extensibility to accommodate new data sources. This capability represents a crucial step towards realizing a polystore system capable of querying data on-the-fly, without the need for prior transformations which is the core of the ELT (Extract-Load-Transform) principle inherent to DLss.

4 DEMONSTRATION

The demonstration[1] applies SEDAR to two use cases. First, we show the process of ingesting datasets, annotating them with semantic labels, integrating them with *PLASMA* and finally joining them using OBDA. The datasets and ontology are from SemTab 2019 [19] and contain data about Countries, their capitals and currencies. Each dataset is ingested into a different internal database (*HDFS*, *Postgres*, *MongoDB*). We integrate them by a semantic model assisted by predictions from our semantic modeling component. Finally, we showcase a SPARQL query to perform a join between them.

The second use case comes from a smart factory project at our research lab, where high-throughput experiments to improve chemical formulations are conducted. Data are available from a REST-API delivering information in JSON format. To find out which chemicals have been used in the latest experiments, the information needs to be joined with the project's relational database, that contains information about the chemicals, using the graphical transformation workflow. Searching for source and metadata, processed datasets as well as semantics is demonstrated next. In the meantime, the relational database, that has been ingested as a DeltaTable, has changed and an update is required. Finally, we showcase MLOps by running a ML experiment.

ACKNOWLEDGMENTS

For implementing the prototype, the authors acknowledge the support of A. Martin, M. Thiel, R. Kuller, L. Beer, F. Lentzen, F. Bongartz, M. Noman, T. Claas, M. Fallouh, Z. Abdullah gratefully. This work has been sponsored by the German Federal Ministry of Education and Research, Germany in the funding program "Forschung an Fachhochschulen", project *I²DACH* (grant no. 13FH557KX0).

[1] Source code and video available at https://github.com/hsnr-data-science/SEDAR

REFERENCES

[1] Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., and Lehmann, J. Pykeen 1.0: a python library for training and evaluating knowledge graph embeddings. *The Journal of Machine Learning Research 22*, 1 (2021), 3723–3728.

[2] Alla, S., and Adari, S. K. *What Is MLOps?* Apress, Berkeley, CA, 2021, pp. 79–124.

[3] Armbrust, M., and et. al. Delta lake: High-performance ACID table storage over cloud object stores. *Proc. VLDB Endow.* (2020).

[4] Beck, D. A. C., Carothers, J. M., Subramanian, V. R., and Pfaendtner, J. Data science: Accelerating innovation and discovery in chemical engineering. *AIChE Journal 62*, 5 (2016).

[5] Brauner, P., Dalibor, M., Jarke, M., Kunze, I., Koren, I., Lakemeyer, G., Liebenberg, M., Michael, J., Pennekamp, J., Quix, C., Rumpe, B., van der Aalst, W. M. P., Wehrle, K., Wortmann, A., and Ziefle, M. A computer science perspective on digital transformation in production. *ACM Trans. Internet Things 3*, 2 (2022), 15:1–15:32.

[6] Deng, D., Fernandez, R. C., Abedjan, Z., Wang, S., Stonebraker, M., Elmagarmid, A. K., Ilyas, I. F., Madden, S., Ouzzani, M., and Tang, N. The data civilizer system. In *Cidr* (2017).

[7] Deng, X., Sun, H., Lees, A., Wu, Y., and Yu, C. Turl: Table understanding through representation learning. *ACM SIGMOD Record 51*, 1 (2022), 33–40.

[8] Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., and Van de Walle, R. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow 1184* (2014).

[9] Ehrlinger, L., and Wöss, W. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS) 48*, 1-4 (2016), 2.

[10] Eichler, R., Giebler, C., Gröger, C., Schwarz, H., and Mitschang, B. Handle-a generic metadata model for data lakes. In *International Conference on Big Data Analytics and Knowledge Discovery* (2020), Springer, pp. 73–88.

[11] Farid, M., Roatis, A., Ilyas, I. F., Hoffmann, H.-F., and Chu, X. Clams: Bringing quality to data lakes. In *Proc. SIGMOD* (2016).

[12] Futia, G., Vetrò, A., and Martin, J. C. D. Semi: A semantic modeling machine to build knowledge graphs with graph neural networks. *SoftwareX 12* (2020), 100516.

[13] Gottschalk, S., and Demidova, E. Tab2kg: Semantic table interpretation with lightweight semantic profiles. *Semantic Web*, Preprint (2022), 1–27.

[14] Hai, R., Geisler, S., and Quix, C. Constance: An intelligent data lake system. In *Proc. SIGMOD* (2016).

[15] Halevy, A. Y., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S., and Whang, S. E. Managing google's data lake: an overview of the goods system. *IEEE Data Eng. Bull.* (2016).

[16] Heyvaert, P., De Meester, B., Dimou, A., and Verborgh, R. Declarative Rules for Linked Data Generation at your Fingertips! In *Proceedings of the 15th ESWC: Posters and Demos* (2018).

[17] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. d., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. Knowledge graphs. *ACM Computing Surveys (CSUR) 54*, 4 (2021), 1–37.

[18] Jarke, M., and Quix, C. On warehouses, lakes, and spaces: The changing role of conceptual modeling for data integration. In *Conceptual Modeling Perspectives* (2017).

[19] Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., and Srinivas, K. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17* (2020), Springer, pp. 514–530.

[20] Khan, Y., Zimmermann, A., Jha, A., Gadepally, V., d'Aquin, M., and Sahay, R. One size does not fit all: querying web polystores. *Ieee Access 7* (2019), 9598–9617.

[21] Labio, W., and Garcia-Molina, H. Efficient snapshot differential algorithms for data warehousing. In *Proc. 22th Intl. Conf. on Very Large Data Bases (VLDB)* (Mumbai (Bombay), India, 1996), Morgan Kaufmann, pp. 63–74.

[22] Mami, M. N., Graux, D., Scerri, S., Jabeen, H., Auer, S., and Lehmann, J. Squerall: Virtual ontology-based access to heterogeneous and large data sources. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II* (2019), C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds., vol. 11779 of *Lecture Notes in Computer Science*, Springer, pp. 229–245.

[23] Paulus, A., Burgdorf, A., Pomp, A., and Meisen, T. Recent advances and future challenges of semantic modeling. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)* (2021), IEEE, pp. 70–75.

[24] Paulus, A., Burgdorf, A., Puleikis, L., Langer, T., Pomp, A., and Meisen, T. Plasma: Platform for auxiliary semantic modeling approaches. In *ICEIS (2)* (2021), pp. 403–412.

[25] Sawadogo, P., and Darmont, J. On data lake architectures and metadata management. *JJIS* (2021).

[26] Sawadogo, P. N., Scholly, E., Favre, C., Ferey, E., Loudcher, S., and Darmont, J. Metadata systems for data lakes: models and features. In *European conference on advances in databases and information systems* (2019), Springer, pp. 440–451.

[27] Schelter, S., and et. al. Unit testing data with deequ. In *SIGMOD Conference* (2019), ACM.

[28] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15* (2018), Springer, pp. 593–607.

[29] Scholly, E., Sawadogo, P., Liu, P., Espinosa-Oviedo, J. A., Favre, C., Loudcher, S., Darmont, J., and Noûs, C. Coining goldmedal: a new contribution to data lake generic metadata modeling. *arXiv preprint arXiv:2103.13155* (2021).

[30] Stalmach, U. Wie funktioniert eine smart paint factory? *Journal of Oberflaechentechnik 59* (2019), 68–73.

[31] Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C., and Tan, W.-C. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data* (2022), pp. 1493–1503.

[32] Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., and Zakharyaschev, M. Ontology-based data access: A survey. International Joint Conferences on Artificial Intelligence.

[33] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull. 41*, 4 (2018), 39–45.

[34] Zhang, G., Schmitz, C., Fimmers, M., Quix, C., and Hoseini, S. Deep learning-based automated characterization of crosscut tests for coatings via image segmentation. *Journal of Coating Technology and Research 19* (2022), 671–683.

[35] Zhao, Y., Megdiche, I., and Ravat, F. Data lake ingestion management. *arXiv preprint arXiv:2107.02885* (2021).